Position Paper
Operating System and Runtime Software for Exascale Systems

Enhanced Resource Management Support for Failure Management

Morris Jette and Danny Auble, SchedMD LLC
<jette, da>@schedmd.com

From a workload management perspective, failure management currently means recognizing a failure after it occurs and restarting the affected job(s), typically using a previously created checkpoint. The user is typically responsible for creating checkpoint images and either allocating extra resources to replace failed components or restarting with a new job allocation. This mode of operation has a severe impact upon application performance if failures are common, which is inevitable on exascale system. We propose developing a collaborative approach to minimize the impact of failures upon applications including active communications between the applications and the resource manager.

Much research has been conducted in the area of failure prediction with some success. This information can be used to drain resources of new work, but a more active approach would be beneficial for jobs currently using those resources. We envision the ideal response to both observed and anticipated failures will notify the application via Remote Procedure Call (RPC) of its details including effected resource, probability of failure, time frame, etc. This information will permit a response best suited to each application at that particular point in its execution. A mechanism for the application to notify the resource manager of failures that it has observed could also be valuable, although some research will be required to assess the accuracy of user reported failures.

A few resource managers support the ability to add resources to running jobs, but we are aware of none which maintain hot spare resources that can be made quickly available to running jobs in response to failed resources. The typical mode of operation today is for each long-running job to allocate extra resources for its exclusive use. This is wasteful of resources and lacks a mechanism to replenish the job's pool of hot spare resources as needed. We believe a better solution is for the resource manager to maintain a common pool of hot spare resources to be made available to applications via RPCs and replenished as needed. In response to an application's request for additional resources, the resource manager may refuse the request, provide replacement resources immediate or provide them at some point at specific time the future.

Extending a running job's time limit is widely supported by resource managers, but generally only available to system administrators or privileged users. However granting regular users the ability to increase a job's time limit in response to failed resources should prove extremely valuable. The addition time may reflect the application's addition execution time with fewer resources than originally allocated and/or the delay incurred waiting for replacement resources.

Challenges addressed: The estimated mean time between failure on exascale systems are well under one hour and well under the execution time of many applications. Effective utilization

of exascale computers requires a robust execution environment including the resource management infrastructure to support jobs surviving multiple failures, which is currently not available.

A key part of this infrastructure is a communication channel between the application and resource manager to better manage failures. Communications are anticipated to include:

1. Notification of the resource manager by the application of observed failures.
2. Notification of the application by the resource manager when resource failures are observed or anticipated.
3. Application request for resources to replace failed components in its current allocation.
4. Application request to the resource manager for an extended time limit, typically reflecting the delay induced loss of resources.

Maturity: The resource manager plays a key role in failure management for exascale systems including the allocation of resources to jobs, enforcing time limits, and knowledge about known and anticipated resource failures. These responsibilities place the resource manager in a key role for failure management. The work described here is only one component of a comprehensive strategy to manage failures, which must include also work on applications and checkpoint/restart mechanisms.

Uniqueness: The importance of failure management is particularly acute on exascale systems due to job run times well in excess of the system's mean time to failure. These techniques could also be useful for systems with lower failure rates, but providing critical services.

Novelty: The work currently underway to improve fault tolerance with the Coordinated Infrastructure for Fault Tolerance Systems (CiFTS) initiative and at the application level are unquestionably valuable. However these efforts have largely excluded the resource manager, which we believe severely impacts their utility at exascale.

Applicability: While failure management is of particular concern to exascale computing, it would be of value to smaller systems with long running jobs and computers operating in hostile environments.

Effort: The effort to effectively explore this approach would be on the order of 2.5 person months. This would represent the effort to more fully explore the benefits of failure management in a resource manager and architect the infrastructure, but exclude any development work.